Algorytmy numeryczne - zadanie 2
Operacje na macierzach

Artur Pyśk
Aleksander Szewczak
grupa 2 - Aplikacje internetowe i bazy danych

18 listopada 2018r.

Sprawozdanie pokazuje analizę poprawnościową oraz wydajnościową algorytmu eliminacji Gaussa w następujących wariantach:

G: bez wyboru elementu podstawowego

PG: z częściowym wyborem elementu podstawowego

FG: z pełnym wyborem elementu podstawoewgo

Testy zostały przeprowadzone na komputerze Lenovo IdeaPad 510-15ISK wyposażonym w procesor Intel Core i5-6200 (2,4 GHz), pamięć 8GB RAM DDR3 i system Windows 10. Testując algorytm używalimy następujących typów reprezentujących liczbę rzeczywistą:

TF: typ pojedynczej precyzji: float

TD: typ podwójnej precyzji: double

TC: typ własny: Fraction

Program, który napisaliśmy w języku C# rozwiązuje układ równań $A \cdot X = B$ dla losowej macierzy kwadratowej A i wektora B. Jako współczynniki macierzy A i wektora X wzięliśmy pseudolosowe liczby zmiennoprzecinkowe z przedziału [-1,1) otrzymane jako iloraz $\frac{x}{2^{16}}$, gdzie x jest pseudolosową liczbą całkowitą z przedziału $[-2^{16},2^{16}-1]$. Wektor B obliczamy jako $B := A \cdot X$. Macierz A i wektor B podajemy jako parametry do rozwiązania układu równań, wektor X zachowujemy jako rozwiązanie wzorcowe.

Aby udowodnić, że algorytm Gaussa został zaimplementowany poprawnie, porównaliśmy wyniki naszego programu dla typu float wraz z wynikami obliczonymi przez WolframAlpha dla macierzy o rozmiarze 2x2.

Macierz $A = \begin{bmatrix} -0.006820267871 & 0.958740234 \\ -0.7615204 & 0.996871948 \end{bmatrix}$, Wektor $B = \begin{bmatrix} 0.763859749 \\ 0.603664458 \end{bmatrix}$

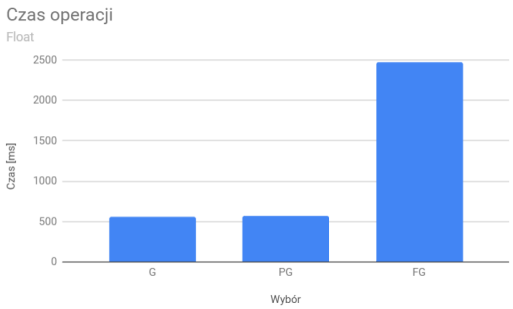
Zestawienie wyników:

Wolfram	G	PG	FG
$\begin{bmatrix} 0.573548 \\ 0.0200814 \end{bmatrix}$	$\begin{bmatrix} 0.573547363 \\ 0.0200805664 \end{bmatrix}$	$\begin{bmatrix} 0.5735474 \\ 0.0200805664 \end{bmatrix}$	$\begin{bmatrix} 0.5735474 \\ 0.0200814 \end{bmatrix}$

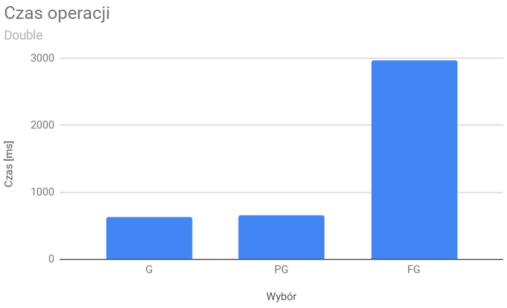
Jak można zauważyć, wyniki są niemal identyczne, natomiast powstaje niewielki błąd, który jest spowodowany nieprecyzyjnością typu zmiennoprzecinkowego.

1 Hipotezy

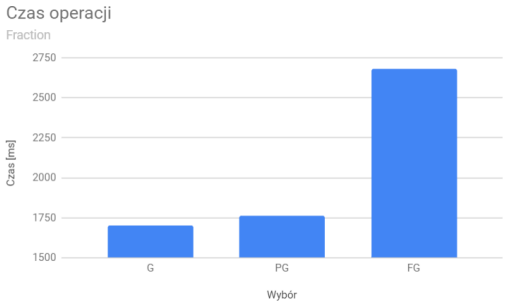
H1: Dla dowolnego ustalonego rozmiaru macierzy czas działania metody Gaussa w kolejnych wersjach (1, 2, 3) rośnie.



(a) Float ($N = 500$)



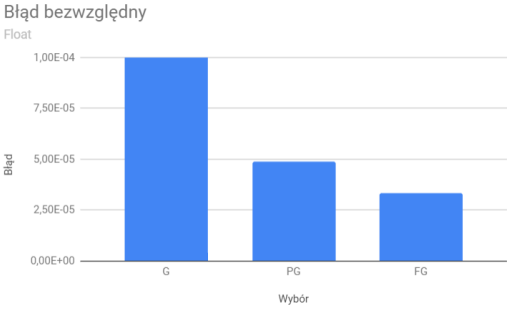
(b) Double ($N = 500$)



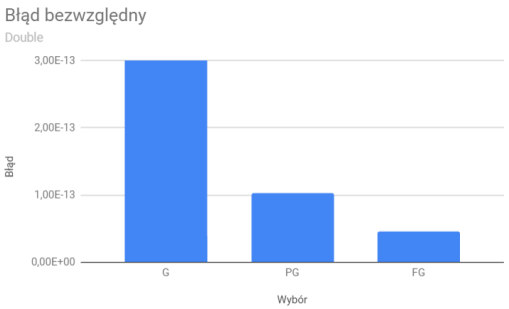
(c) Fraction ($N = 50$)

Jak możemy zauważyć na powyższych wykresach, zarówno dla typu Float jak i Double, dla $N = 500$ oraz typu Fraction, dla $N = 50$, czas działania metody Gaussa w kolejnych wersjach (1 ,2, 3) rośnie. Przy kilkunastu próbach dostrzegliśmy, że nie ma dużej różnicy miedzy wariantami G i PG. Co jest zastanawiające, w kilku przypadkach czas wariantu PG jest mniejszy niż G. Prawdopodobine jest to spowodowane tym, że w kilku testach PG został uruchomiony później niż G. Wariant FG jest zdecydowanie najwolniejszy. Okazuje się, że hipoteza jest prawdziwa dla większości przypadków. W zależności od ustalonego N algorytm PG wykonuje się nieznacznie dłużej niż G. Wariant FG zawsze trwa największą ilość czasu.

H2: Dla dowolnego ustalonego rozmiaru macierzy błąd uzyskanego wyniku metody Gaussa w kolejnych wersjach (1, 2, 3) maleje.



(a) Float



(b) Double

Jak zauważamy na powyższym wykresie, zarówno dla typu Float jak i Double, dla $N = 500$, błąd uzyskanego wyniku metody Gaussa w kolejnych wersjach (1, 2, 3) maleje. W przypadku obu typów, największy błąd następuje przy wariancie G. Błędy wariantów PG i FG są sobie bliższe, natomiast największą dokładność ma algorytm z pełnym wyborem. Okazuje się, że hipoteza jest prawdziwa dla wszystkich przypadków. Najbardziej korzystny jednak wydaje się algorytm Gaussa z częściowym wyborem biorąc pod uwagę stosunek jego dokładności do wydajności. Wykonuje się sporo krócej, niż wariant FG, natomiast daje zbliżone wyniki. Dla własnego typu Fraction, błąd dla każdego wariantu jest taki sam - równy 0. Dlaczego tak się dzieje, wyjaśniamy w poniższej hipotezie.

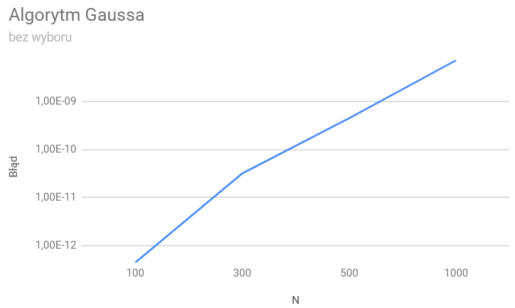
H3: Użycie własnej arytmetyki na ułamkach zapewnia bezbłędne wyniki niezależnie od wariantu metody Gaussa i rozmiaru macierzy.

n	B(G)	B(PG)	B(FG)	T(G)	T(PG)	T(FG)
10	0	0	0	23	24	29
20	0	0	0	309	321	491
30	0	0	0	1293	1353	1705
40	0	0	0	4298	4613	5698
50	0	0	0	10865	11492	14840

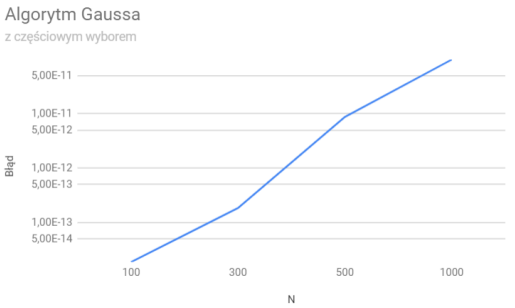
Patrząc na powyższą tabelę prezentującą własny typ Fraction, gdzie B - błąd bezwzględny, T - czas [ms] dochodzimy do wniosku, że użycie własnej arytmetyki eliminuje błąd obliczeń. Dzięki wykorzystaniu typu BigInteger, który nie posiada żadnego limitu przechowywania danych (jedynym ograniczeniem jest nasz komputer) możemy wykonywać skomplikowane obliczenia bez błędów. Hipoteza jest więc prawdziwa. Użycie własnej arytmetyki na ułamkach zapewnia bezbłędne wyniki niezależnie od wariantu metody Gaussa i rozmiaru macierzy. Z powodu ograniczeń sprzętowych, ciężko było nam zrobić odpowiednie testy dla większych macierzy.

2 Pytania

Q1: Jak zależy dokładność obliczeń (błąd) od rozmiaru macierzy dla dwóch wybranych przez Ciebie wariantów metody Gaussa gdy obliczenia prowadzone są na typie podwójnej precyzji (TD)?



(a) Bez wyboru

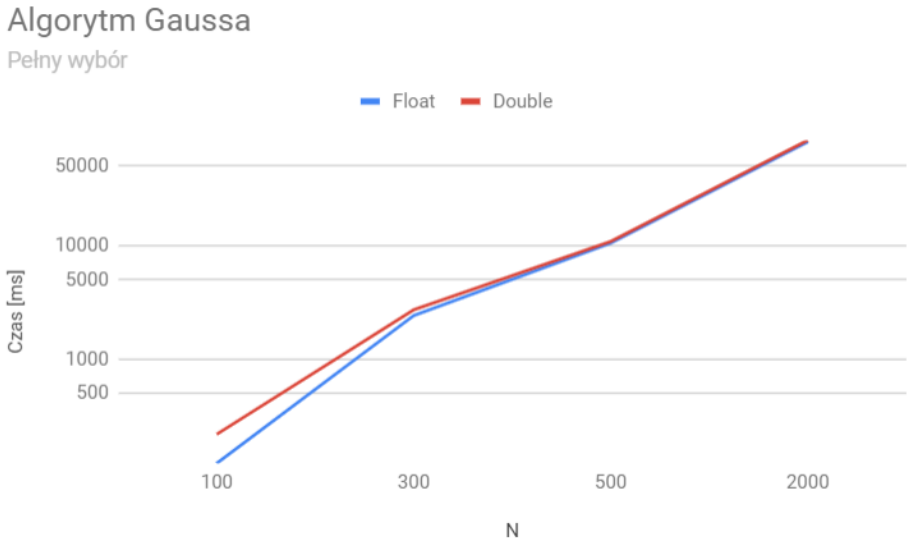


(b) Z częściowym wyborem

Do testów, które umożliwiły nam odpowiedź na to pytanie skorzystaliśmy z wariantów G oraz PG. Jak możemy zauważyć na powyższym wykresie, im większa macierz, tym błąd rośnie i wynik staje się mniej dokładny. Dokładność obliczeń maleje wraz z wzrostem rozmiaru macierzy.

Q2: Jak przy wybranym przez Ciebie wariancie metody Gaussa zależy czas działania algorytmu od rozmiaru macierzy i różnych typów?

Do testów, które umożliwiły nam odpowiedź na to pytanie skorzystaliśmy z wariantu FG. Jak możemy zauważyć na poniższym wykresie oraz tabeli z hipotezy trzeciej, im większa macierz tym dłuższy czas wykonywania algorytmu, niezależnie od typu. Typ double wykonuje się trochę dłużej niż typ float. Najdłuższy czas działania ma jednak własny typ Fraction.



3 Wydajność implementacji

Typ	T(G)	T(PG)	T(FG)
n = 500			
Float	554ms	569ms	2473ms
Double	633ms	651ms	2966ms
n = 100			
Fraction	246179ms	247009ms	322440ms

Z powodu ograniczeń komputera, testowanie typu z własną arytmetyką ograniczyliśmy do macierzy dla N = 100. Zdecydowanie najdłużej wykonuje się typ własnej arytmetyki. Typ double wykonuje się trochę dłużej niż float, ale nie ma ogromnej różnicy.

4 Podział pracy

Aleksander Szewczak	Artur Pyśk
Praca nad strukturą projektu	Praca nad strukturą projektu
Algorytm - warianty PG i FG	Algorytm - wariant G
Przeprowadzenie testów	Implementacja klasy generycznej MyMatrix
Generowanie wykresów	Implementacja typu własnej precyzji Fraction
Przygotowanie sprawozdania	Obliczanie błędu